IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 29, 2020, accepted July 7, 2020, date of publication July 13, 2020, date of current version July 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3008523

Self-Supervised Animation Synthesis Through Adversarial Training

CHENG YU[1,2], WENMIN WANG[1,3], (Member, IEEE), AND JIANHAO YAN[1]
[1]Faculty of Information Technology, Macau University of Science and Technology, Taipa 999078, China
[2]Chongqing College of Electronic Engineering, Chongqing 401331, China
[3]International Institute of Next Generation Internet, Macau University of Science and Technology, Taipa 999078, China

Corresponding author: Wenmin Wang (wmwang@must.edu.mo)

This work was supported by the Science and Technology Development Fund (FDCT) of Macau under Grant 0016/2019/A1.

ABSTRACT In this paper, we propose a novel deep generative model for image animation synthesis. Based on self-supervised learning and adversarial training, the model can find labeling rules and mark them without origin sample labels. In addition, our model can generate continuous changing images based on the automatically labels learning. The labels learning model can be implemented on a large number of out-of-order samples to generate two types of pseudo-labels, discrete labels and continuous labels. The discrete labels can generate different animation clips, and the continuous labels can generate different frames in the same clip. Embedding pseudo-labels with latent variables into latent space, our model discovers regularities and features from latent space. Animation features are fully characterized by the pseudo-labels learned from the self-supervised module. Using upgraded adversarial training steps, the model learns to map animation features to pseudo-labels from the latent space and then organizes pseudo-labels embedding into latent variables to generate animation features. By adapting dimensions of pseudo-labels, we match fine features with latent variables. Such as using the two types of pseudo-labels, our model can also generate different styles of videos from the same dataset. The specific implementation tricks depend on the different pseudo-label dimensions and the number of pseudo-label dimensions. Comparing the results of our model with other state-of-the-art approaches, the model does not use complicated components, such as 3D convolution layers and recurrent neural networks. Our experimental results show that an appropriate number of the pseudo-label dimensions can better characterize animation features. In this case, an animation which reached human-level perception can be synthesized. The performance of animation synthesis has reached relatively superior results on several challenging datasets.

INDEX TERMS Self-supervised learning, animation synthesis, pseudo-label, adversarial training.

I. INTRODUCTION

Visual generation is a frontier challenge in computer vision, and it can be used to create visual content automatically and so on. By training out-of-order image samples, generating an animation from noisy images or a single image is a complex problem.

The challenge of visual generation can be briefly divided into two tasks. The first task is image generation, which focuses on different image synthesis applications, such as image to image translation [1], [2], high resolution image synthesis [3]–[7].

The second task is video generation, which involves applying trained a model to generate video. Comparing with image generation, video generation needs not only to generative images that treated as frames from a training video, but also to generate a changing and continuous sequence that corresponded with the original frames.

There are two subtasks for video generation. One of them is video prediction [8], [9]. This task trains video clips to predict a few future frames. Another one is animation synthesis [1], [10], [11]. By training a single image or random image samples, the task needs to make a model that generates a short image sequence.

All of the above work depends on generating models. Current mainstream generative models include Generative Adversarial Networks (GAN) [12], Variational Auto-Encoders (VAE) [13], Autoregressive models [14], [15], and Flow models [16]. In this article, adversarial training mainly refers to training the networks of GAN. GAN is one of the most popular models that can generate high quality and quantity image, it is widely used in image synthesis tasks.

However, GAN still has many difficulties and challenges when directly generating video. Because GAN is used for

The associate editor coordinating the review of this manuscript and approving it for publication was Shiping Wen.

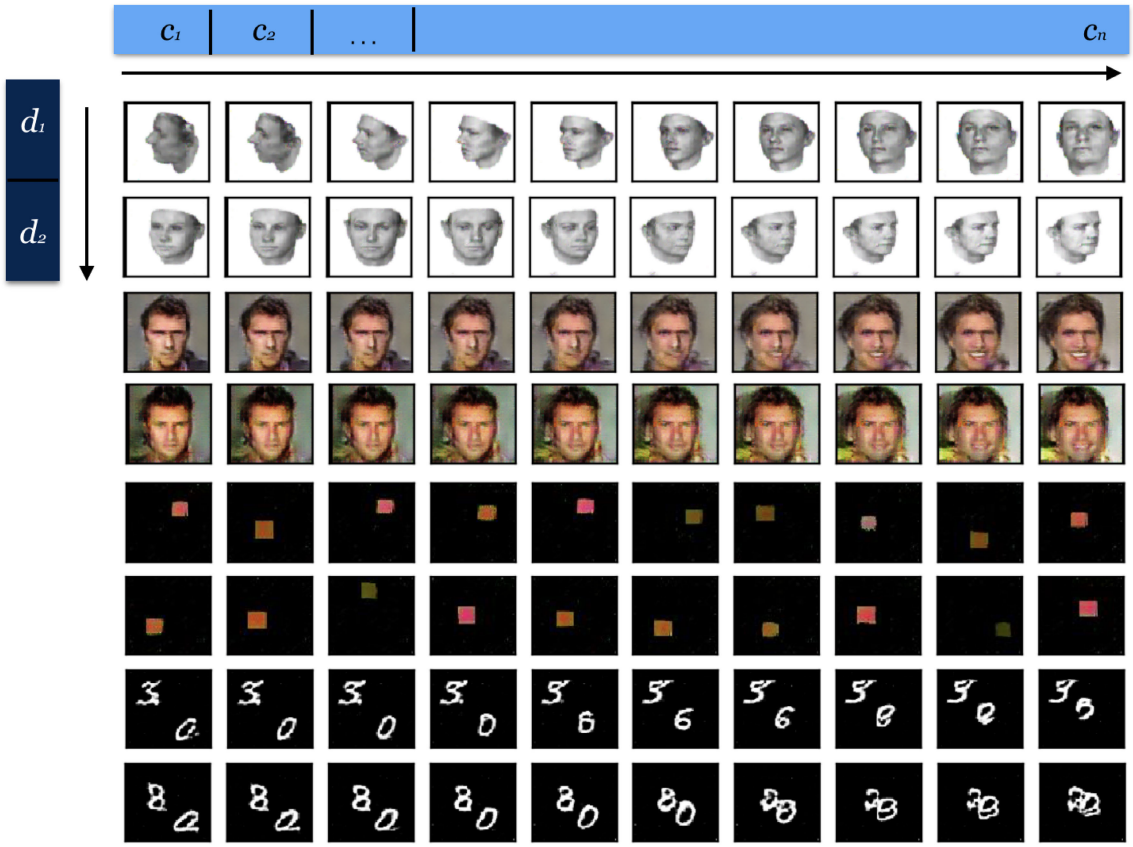

FIGURE 1. Our deep generative model can synthesize an animation from shuffled images. Using a self-supervised manner, the model learned discrete labels $\{d_1, d_2\}$ and continuous labels $\{c_1, c_2, \ldots, c_n\}$ without any given labels. Discrete labels $\{d_1, d_2\}$ represent different animation clips. Continuous labels $\{c_1, c_2, \ldots, c_n\}$ represent different motions in one type of clips. Finally the model can generate motions by learned d_i and c_i from diverse datasets.

generating an image as a single frame. Besides, the task of video generation also needs to generate motions related to the content of consecutive frames. The main method currently used for processing image sequences is the Recurrent Neural Network (RNN) [17]–[19], which is widely used in natural language processing to generate text sequences. But in the training process, RNN requires a large number of known sequences as prior conditions for generating new sequences. the RNN node should share its weights with other nodes in the same sequence. So the main disadvantage of RNN is that generating sequences requires a large number of known sequences for calculation, especially in image sequence generation [20].

Current research rarely generates animations using a single GAN directly. In a semi- supervised manner, these methods should take sequence information to generate consecutive frames, such as using 3D Convolutional Neural Networks (Conv3D) [21]–[23], GAN [24], [25], and RNN [11] to generate motions, at next, using other networks to generate each content frame.

In a supervised manner, the current works are mainly utilizing appropriate prior conditions as labels [1], [2], [10], [26] to control each frame generation, such as Conditional Generative Adversarial Networks (CGAN) [27]. The training of CGAN concatenates labels with the label's image samples. The semantic segmentation and optical flow can be used as prior conditions that treat as image sample labels. Using different types of labels, the target images can be generated according to the corresponding labels. Especially using a semantic segmentation as the label, the label can control the image generation at the pixel level. But this method usually needs large-scale prior conditions as image labels. Meanwhile, the method always results in pixel distortions in some areas of its generated images.

We design a model that can learn two types of pseudo-labels by itself. Different from the real label, The pseudo-labels are obtained through image features calculation. By implementing the self-supervised training module, the model outputs animation from random image samples. Our model does not require any real labels and sequences. The result of our animation synthesis can reach the human level perception on multiple datasets. Some examples are given in Fig. 1.

Our contributions are summed up as follows:

(1) We propose an animation synthesis model based on adversarial training. This model can learn two types of pseudo-labels in a self-supervised manner and embed the pseudo-labels to latent variables, so that the model can map

the pseudo-labels to animation features which come from latent space.

(2) We show the interrelationships between the two types of pseudo-labels, and the different results by different dimensions of pseudo-labels. One type of pseudo-label (discrete label) can generate different types of animation clips. The other kind of pseudo-label (continuous label) can generate different frames in the same animation clip. We combine two types of pseudo-labels to complete animation synthesis

(3) By improving existing image quality metrics, we propose a new method to measure the quality of generated animations. Our method takes sequence continuity into account when evaluating generated images. Compared with the current state-of-the-art methods, our experiment results verify the effectiveness of our proposed method.

II. RELATED WORK

A. GENERATIVE ADVERSARIAL NETWORKS

GAN is a generative model based on the minimax game theory. Since its first appearance in 2014, it makes a huge success in image generation tasks. The original GAN [12] is an unsupervised generation model, composing of two networks, Generator and Discriminator.

We denote Generator as G and Discriminator as D. In practice, the brief loss-function as

$$\max_G \min_D f(D, G). \tag{1}$$

The G inputs random noise $z \sim p_z$, and then generate fake image samples $G(z)$. The D inputs both real and fake samples and tries to discriminate them.

We use back-propagation to update the parameters of the networks, training the networks to optimize our target function. The first step is that we should train D to figure out the fake image samples, as

$$f(D)_1 = -\mathbb{E}_{z\sim p_z}[log(1 - D(G(z)))], \tag{2}$$

and then we denote the true image samples as $x \sim p_x$, try to figure out the true image samples as follow

$$f(D)_2 = -\mathbb{E}_{x\sim p_x}[logD(x)]. \tag{3}$$

At the next step, we train G depending on D. Let D guide G to generate the better fake image samples, the process is given by

$$f(G) = -\mathbb{E}_{z\sim p_z}[logD(G(z))]. \tag{4}$$

The GAN is a good strategy to estimate the complicated distribution. However, GAN is not easy for training [28]. In order for GAN to generate higher resolution, better quantity and quality image samples, there are two problems that need to be solved [5], [29]–[32], one is model collapse and the other is training instability.

B. CONDITIONAL GAN AND LATENT VARIABLES

Conditional Generative Adversarial Nets (CGAN) [27] is an extended type of GAN. We denote y as the labels of the image samples x. The CGAN loss function is given by

$$f(D) = -\mathbb{E}_{x\sim p_x}[logD(x|y)] + \mathbb{E}_{z\sim p_z}[-log(1 - D(G(z|y)))]. \tag{5}$$

The entire input of G comes from different distributions in high dimensions, and these input variables are called latent variables. ACGAN [33] and InfoGAN [34] are improvements of CGAN. The ACGAN does not take labels y as input. Instead, it outputs y by D, while adding cross-entropy loss to estimate the output labels y. In addition, InfoGAN adds more parameters to estimate multi labels in an unsupervised manner.

StyleGAN [6] does not use labels as prior conditions, it uses fully connected layers to better embed the latent variables when generating images. So that the StyleGAN embedded latent variables can be used as labels to generate images. However, latent variables used to generate images are usually more complex than known labels, which come from more complex distributions and cannot be directly used as good labels for animation synthesis.

C. ANIMATION SYNTHESIS AND VIDEO PREDICTION

Video prediction and animation synthesis are two subtasks of video generation. Video prediction aims to generate continuous video frames based on previous video frames. Similar to most generation problems of sequence to sequence, RNN [17]–[19] are often used to deal with this problem.

MoCoGAN [11] separates contents and motions from the same video. It trains one type of RNNs, Gate Recurrent Units (GRUs) [19], to generate motions and train GANs to generate content images (as frames). Similar to training RNNs, Uni&Bi [9] generates intermediate sequences by training its front sequence and rear sequence.

During RNNs training, each RNN's node needs to share its weights with other neighbors [20]. The calculation and update of the current node weights depend on the update of many adjacent node's weights. In this way, the input sequences can be fitted to the sequences composing of RNN nodes. However, as the sequence length increases, the requirement of calculation will explode, and meanwhile the quality of the sequence generation will become worse.

Animation synthesis is another subtask of video generation. This direction aims to use image samples to generate video. Common methods include using pose estimation [2], [35], [36], optical flow [10], [37], [38]and semantic segmentation [1], [2] as prior conditions to train model and generate continuous images with corresponding conditional features. Pixel-level control can be achieved by generating the labeled pixel distribution.

By using the prior conditions as training labels, each label needs to be the same size as its paired image sample. The requirement of various detailed labels makes animation

synthesis methods that can only be implemented in a supervised manner.

In addition, TGAN [24] uses a GAN to generate motions. Similar to RNN, The GAN uses sequence information as the weak labels which come from a video dataset. The weak labels come from a video dataset that corresponds to a category of the video. These labels can be used to generate motions by the GAN. Such labels are weak and cannot be controlled to generate pictures and videos in pixel-level.

III. METHOD

Inspired by InfoGAN and StyleGAN, we design a novel model that can learn pseudo-labels as an animation sequence guide in a self-supervised manner.

Our method bases on adversarial training. It not only uses an improved CGAN to estimate common labels but also improves the estimation of conditional variables in the model's input and output. We propose a self-supervised module to make two types of pseudo-labels, discrete labels d_i and continuous labels c_i. As shown in Fig.1, these are two types of pseudo-labels that control the image generation to human-level perception changes.

A. DISCRETE LABEL LEARNING

As show in Eq. (5), y can be treated as the one-hot variables or the pixel-level variables (e.g., from semantic segmentation).

Here we denote discrete labels d as one-hot variables. d_i is one of the dimensions of d which can be inputted to G (as $G(d)$), then the D tries to figure out the better pseudo- labels d and gives a label score λ_d.

The difference between our method with CGAN is that we do not input any ground-truth labels as prior conditions. Instead, we input random discrete variables which combine with latent variables z. We optimize the D to output a judged score which is corresponded to learned pseudo-label. After that, we optimize G to generate images with intermittent changes through discrete labels d.

Finally, the learned discrete labels d will be embedded to the latent variables. D and G jointly use the same loss-function for optimization, this loss-function is as follows:

$$\lambda_d = -\mathbb{E}_{d\sim p_d}[logD(G(d|z))\cdot d]. \tag{6}$$

B. CONTINUOUS LABEL LEARNING

Different from discrete labels d, we define continuous labels c as continuous variables, and one of the dimensions of continuous labels c can be denoted as c_i. Compared with discrete labels d, c will be more sensitive to generative control. So we do not use cross-entropy (CE) like Eq. (6). We use mean-square error (MSE) as the loss-function to optimize continuous labels c. Although both d and c coming from different uniform distributions, optimizing continuous variables depend on one dimension of discrete label d. we denote the loss-function as

$$\lambda_c = \mathbb{E}_{c\sim p_c}(D(G(c|d_i)) - c)^2. \tag{7}$$

Same as d, the learned continuous labels will be embedded to the latent variables which are corresponded with one d_i. The learned c will control the details of the image changes.

C. SELF-SUPERVISED LEARNING MODULE

Here, we introduce a self-supervised module to learn pseudo-labels composing of discrete and continuous labels. During model training, we input latent variables z that come from Gaussian distribution. Meanwhile we input discrete variables embedded discrete labels d and continuous variables embedded continuous labels c.

In Eq. (1), we train a model from z so that the model can output fake images. The method optimizes a network D in a loss-function given by Eq. (2) and Eq. (3), it also optimizes a network G given by Eq. (4).

Next in Eq. (5), we need to know the labels y as prior conditions. But in a self-supervised manner we do not need to know the labels y. We try to learn pseudo-labels by d and c from an improved network D. In practice, we denote variables concatenation as $\oplus$. The D can be defined as

$$x', d', c' = D(G(z \oplus d \oplus c)). \tag{8}$$

The D not only outputs a score x' to judge the quality of the image generated by G, but also outputs two additional scores d' and c' to judge the quality of the discrete variables and the continuous variables. Each type of output score corresponds to that type of learning pseudo-label.

In two types of pseudo-labels, the interval of discrete labels is greater than the interval of continuous labels, so we first optimize Eq. (6) to learn discrete labels d, and then optimize Eq. (7) to learn continuous labels c in the same discrete label. With the improved D, we define its overall optimization as

$$f(D, G) = \mathbb{E}[-log(1 - D(G(z \oplus d \oplus c)))] + \mathbb{E}[-logD(x)] + \lambda_d + \lambda_c. \tag{9}$$

The target function is shown in Eq. (9), we input latent variables z, pseudo-labels d and c at the same time. When processing 3 types of variables from the outputs of D, which shown in Eq. (8). We first process the d', then process c' in one dimension of d'. By training in this way, we make sure the continuity of the generated animation from generator G.

D. TRAINING STEPS

As shown in Fig. 2, we design two stages to train our model. In stage 1, we optimize a discriminator network D and use its outputs, x', d' and c', to evaluate one latent variable z and two types of pseudo-labels which are represented by two types of variables d and c. The latent variables z, which is required in the common unsupervised generative model, comes from Gaussian distribution. Two types of the marked variables, d' and c', are used to learn two levels of latent features. In stage 2, we optimize a generative network G to generate sequences from three types of input variables z, d and c. z sampled from Gaussian Distribution, d sampled from discrete uniform distribution and c sampled from continuous uniform distribution.

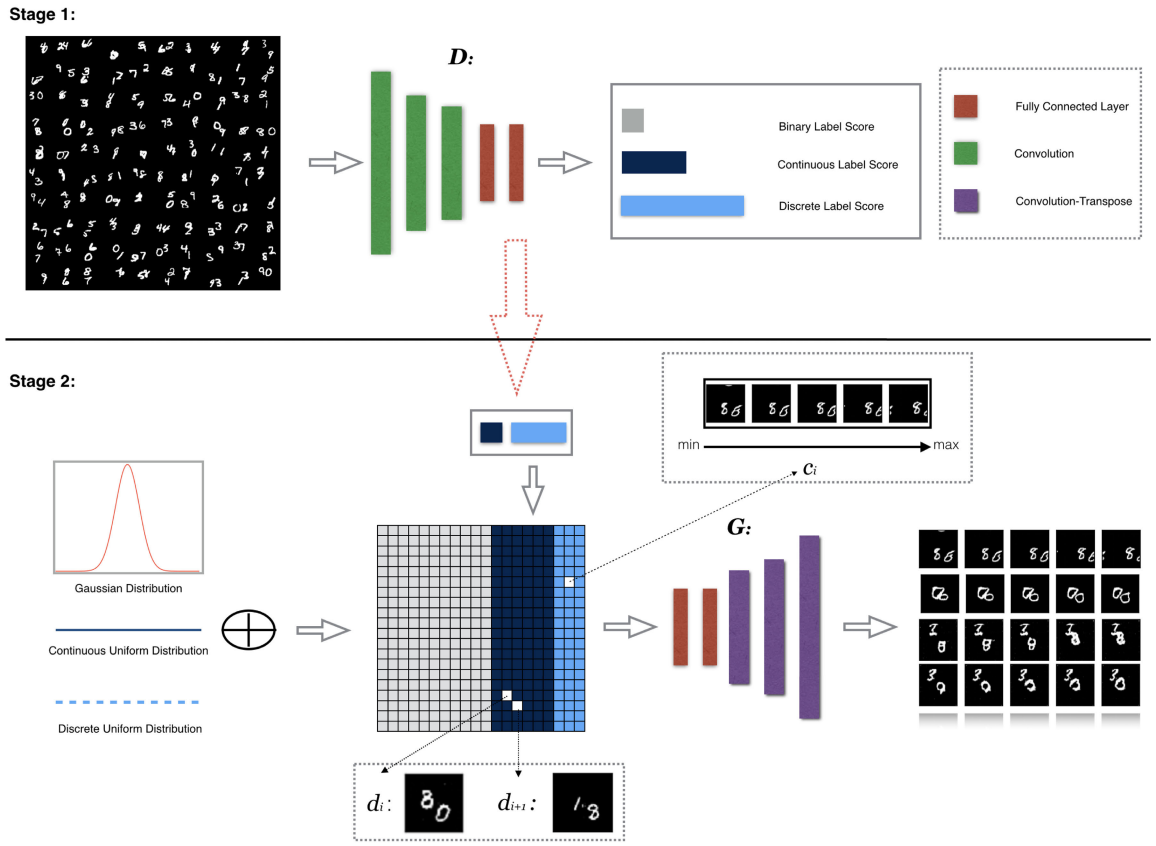

FIGURE 2. The pipeline of our proposed model with adversarial training for animation synthesis. In stage 1, we design a discriminator *D*. *x* is the input which is a batch of disordered images. *D* output three types of label scores that contain discrete label score d' and continuous label score c'. In stage 2, we design a generator *G*. *z*, *d* and *c* are its input, which will be compared with the output from the stage 1, x', d' and c'. We need to train *d* and *c* to be representatives of pseudo-labels. We choose multiple convolution2D layers as the upsampling operation and multiple convolution-transpose2D layers as the downsampling operation. Some repeated layers of the network have been omitted. At the same time, the normalization layers and the activation layers between the sampling layers are also omitted. The above operations implement on the PyTorch library [39]. More detailed network Architecture can refer to Tab. 1 and Tab. 2.

The binary label score x' output from D, we use Binary Cross Entropy (BCE) as the loss function to optimize D. For two different types of pseudo-label scores output from D, discrete label score d' and continuous label score c', we respectively use Cross Entropy (CE) as the loss function of discrete label score and Mean Square Error (MSE) as the loss function of continuous label score. We use two different loss functions to deal with different types of pseudo-labels score to ensure that continuous labels are more sensitive than discrete labels. In the second stage of Fig. 2. We can see the difference in the results of two different dimensions of d_i and numerical changes of a c_i value. When calculating the update parameters in back propagation, we use Adam [40] as the optimizer. Our method are implemented on a deep learning library called PyTorch [39].

E. NETWORK ARCHITECTURE

Tab. 1 detailed the generator G architecture which used in our model. The G we designed is mainly composed of two types of layers, fully connected layers, and convolution-transpose2D layers. They are designed as functions and denoted as Linear() and Deconv2d() respectively. The input

TABLE 1. Network architecture of the animation generator *G*.

G	Configuration
Input	$z_i \sim N(0,1)$, $d_i \sim U\{1, n_d\}$, $c_i \sim U(0,1)$
1	Linear($n_z + n_d + n_c$, 1024), BN, ReLU
2	Linear(1024, 128×8×8), BN, ReLU
3	Reshape(-1, 128, 8, 8)
4	Deconv2d(128, 64, K=4, S=2), BN, ReLU
5	Deconv2d(64, 32, K=4, S=2), BN, ReLU
6	Deconv2d(32, 3 K=4, S=2), BN, Tanh

of G includes three types of variable, z_i, d_i and c_i. The latent variable z_i comes from latent space and follows gaussian distribution. The discrete label d_i follows discrete uniform Distribution and the continuous label c_i follows continuous uniform Distribution. Except for the last layer, all other layers use batch normalization (BN) layer followed by the ReLU nonlinearity. The last layer uses Tanh nonlinearity which is more suitable for output. We denote the parameter of kernel size as K and the parameter of stride size as S.

As shown in Tab. 2, we show the discriminator D which used in our model. The input of D is a batch of frames.

TABLE 2. Network architecture of the animation discriminator *D*.

D	Configuration
Input	[N, C, H, W]
1	Conv2d(C, 32, K=4, S=2), LeakyReLU
2	Conv2d(32, 64, K=4, S=2), LeakyReLU
3	Conv2d(64, 128 K=4, S=2), BN, LeakyReLU
4	Reshape(-1, 128×8×8)
5	Linear(128×8×8, 1024, K=4, S=2), BN, LeakyReLU
6	Linear(1024, $n_z + n_d + n_c$, K=4, S=2)

We record the number of frames as N, the number of channels in one frame as C, and the height and width of the frame as H and W respectively. In *D*, we mainly use the convolution2D layer and denoted it as Conv2d(). Different from the *G*, we only use BN for the last Conv2d layer. And the last fully connected layer does not need BN layer and nonlinearity. The above changes are to make the *D* better to output the evaluation of the three types of variables.

The addition of fully connected layers can enhance the ability of latent variable representation. Here, we use two fully connected layers to enhance the ability of *G* and *D*.

IV. EXPERIMENTS

In this part, we first introduce 5 datasets that we use for our experimental benchmarks. Next, we show an ablation study about the numbers of the pseudo-label dimensions. After that, we introduce a new approach to evaluate the quality of animation synthesis. Finally we compare our model with existing related models by multiple metric methods.

A. BENCHMARK DATASETS

1) 2D SHAPES

The dataset contains 4,000 videos [42], and each video involves different 2D moving shapes. There are three kinds of shapes (triangles, circles and squares) with different sizes and colors. We split the video as clips, and we choose 15,997 clips contain moving circles and squares. Each clip contain 32 frames and the size of each frame is 64×64 pixels. We do not need any sequence from the dataset. So we split the clips to individual frames(256,002 frames in total) and shuffle these frames as random images.

2) MOVING MNIST

The Moving-MNIST [43] contains 10,000 sequences. Each of sequence have 20 frames showing 2 handwritten digits moving in a 64×64 pixel frame. In this experiment, we split this sequences to individual frames. Finally, we choose 9,000 sequences to make 180,000 shuffled frames as our training input images.

3) 3D BASEL FACE MODE

The 3D Basel Face Model (BFM) [41] is calculated from registered 3D scans of 100 male and 100 female faces. Each face presents the main characteristics of the head, gender, and facial features of the person. We make 40,000 different face images to generate animation of 3D face.

4) HUMAN ACTION

From [44], the human action contain 81 clips of 9 people performing 9 human action including jumping-jack and waving-hands. By selecting 71 clips, we make the clips to 7,199 frames and shuffle them. In order to keep the same image size with other our selected dataSets, we also scaled the shuffled frames to 64×64 pixels.

5) CelebFaces ATTRIBUTES DATASET

CelebFaces Attributes Dataset (CelebA) [45] is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 202,599 number of face images. The CelebA contains in-the-wild type and align-cropped type. We choose align-cropped type with original image size is 178×218 pixels and crop each image to 64×64 pixels. Unlike using BFM dataset, in our experiments, we focus on generating human 2D facial animations through CelebA.

B. ABLATION STUDY

In this section, we present an ablation study to empirically assess the impact and correlation when we use the different numbers of pseudo-label dimensions. In previous section, we have introduced our self-supervised module that can learn two types of pseudo-label from shuffled samples, discrete labels d and continuous labels c. In our design, d are pseudo-labels that can distinguish different clips which are apart of one animation. The continuous labels c are used to distinguish detailed features, and the features represent different motions in one kind of clips.

But in reality, we can not determine the required dimensions of the two types of pseudo-labels in different datasets. Therefore, the dimensions of d and c are problems that need to be explored. We denote n_d as the number of discrete label dimensions and denote n_c as the number of continuous label dimensions.

We have proved a result by our experiments that if there is large number of n_d and n_c to learn, the difficulty of optimizing will increase and all the features of the samples cannot be found well. Otherwise, if there is a small number of n_d and n_c, the model also cannot be classified well according to the small dimensions, and the training results are also blurred. In order to get a better experimental result, the dimensions of pseudo-labels need to suit the size of the dataset.

In small-scale datasets, such as BFM and Human Action. n_d is suitable to take a smaller number. As shown in Fig. 3, the BFM sequence changes significantly when n_d is small ($n_d = 5$), as n_d increases, the changing performance becomes weak ($n_d = 10, n_d = 20$). Corresponding with n_d, the number of n_c need to be set in a range. As the Fig. 4 shows, in the case of fixed n_d ($n_d = 20$) using Human Action dataset, too small dimensions of c ($n_c = 5$) leads to a poor result that each frame is the same feature, and also the features learned are not obvious when n_c is too large ($n_c = 20$).

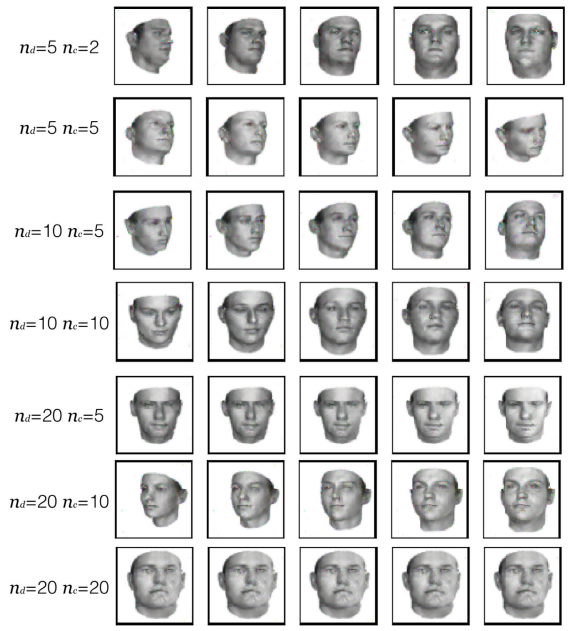

FIGURE 3. Ablation study on 3D Base Face mode (BFM) [41], we compare with the different numbers of the discrete label dimensions (n_d) with different numbers of continuous labels dimensions (n_c). The results show that the growth of n_d will cause the influence of c to become weak. At the same time, as the n_c increases, its influence will become weak or even ineffective (the last row).

FIGURE 4. A part of ablation study on human action [44]. When the gap between n_d and n_c is too large, that will cause the result to blurred or even invalid.

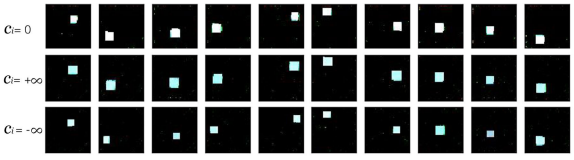

FIGURE 5. A set of experiments for quantitative analysis of c_i in 2D shapes [42] dataset, the same dimensions of discrete labels d and continuous labels c can generate regular sequences, and the difference value of c_i control the color (row 2) or size (row 3) of the object in the animation sequence.

The different d_i causes the generated images to change significantly. Otherwise, the different c_i causes these images to change slightly. One of the reasons is that the value of d_i is the one-hot type and its value set to true or false, while c_i is a continuous variable that can change the value within a range. As shown in the different columns in Fig. 5. The change of c_i is within a range of values. If c_i exceeds this range, the maximum and minimum values will also allow the generated sequence to converge to a stable feature. Our method can set the maximum and minimum values of c_i

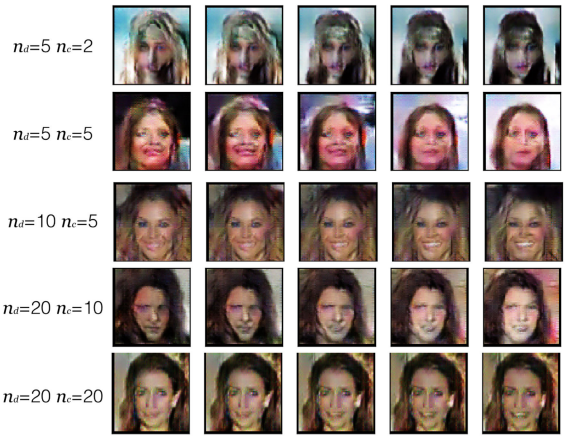

FIGURE 6. Ablation study on CelebFaces Attributes Dataset(CelebA) [45], in this large-scale dataset, we found that the animation generation results are blurred when n_d and n_c are too small. As the n_d and n_c increased, the animation synthesis performance will be better. At the same time, an appropriate increase in n_c will make the animation control more detailed and diverse.

TABLE 3. Ablation study on datasets of different sizes.

n_d, n_c	FID(BFM)	FID(CelebA)
n_d=5 , n_c=2	4.24	11.24
n_d=5 , n_c=5	**2.33**	10.55
n_d=10 , n_c=5	3.65	9.42
n_d=20 , n_c=10	4.07	**6.67**
n_d=20 , n_c=20	5.86	17.10

to generate objects of different colors (row 2) or shapes (row 3) in the same sequence.

In large-scale dataset (such as CelebA). n_d is suitable to take a larger value. As shown in Fig. 6, the quality of the CelebA sequence generated by our method is significantly improved when n_d is greater than 5. At the same time, with the increase of n_c, the performance of image change in the same value of d_i will be more detailed and diverse (from the 3rd row to the 5th row).

We provide a comparison of the ablation study on two datasets showing in Tab. 3. When n_d and n_c are smaller ($n_d = 5$, $n_c = 5$), the small-scale dataset BFM works better. Large-scale datasets CelebA requires larger n_d and n_c ($n_d = 20, n_c = 10$). We use FID [46] (low is better) to measure the quality of the generated sequence.

The embedded pseudo-labels follow two uniform distributions (discrete and continuous). Compared with the original latent variables that follow the Gaussian distribution, the pseudo-labels simplifies the mapping of animation features to latent variables. The one reason is that the uniform distribution is easier to control. In addition, the discrete uniform distribution and the continuous uniform distribution also easy to make a sizable difference.

Two types of uniform distributions also provide convenience for learning the two types of pseudo-labels and forming distinct features mapping. In the animation features, the different clips correspond to discrete variables, and different frames

in the same clip correspond to continuous variables. Both types are implicitly embedded as pseudo-labels in the latent variables that follow the Gaussian distributions and reduced the mapping complexity on latent space to animation features.

C. METRICS OF ANIMATION EVALUATION

In the image quality evaluation method, Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are two well-known objective image quality metrics [47], [48].

PSNR measures the quality of the picture by the MSE between the pixels compared with the ground-truth. The larger the PSNR value, the better the generated image quality, and the smaller distortion. SSIM compares the generated image luminance, contrast and picture structure with ground-truth. When the generated image is the same as ground-truth, SSIM gets the maximum value. Both methods measure a single image and cannot measure a continuous sequence of animation.

In the method of measuring the image generation model, Inception Score (IS) [49] and Fréchet Inception distance (FID) [46] are two of the most popular methods, which are used to measure the distance between the distribution of the generated images and the distribution of the ground-truth. IS measures the distribution distance between the generation and ground-truth by comparing the distance between them and ImageNet [50], a large-scale hierarchical image database. When the generated distribution to be measured is not similar to ImageNet, IS cannot accurately measure the generated model. FID can directly measure the similarity between the distribution under test and ground-truth, but neither of them can accurately measure the sequence relationship between the image distribution.

To solve the problem of image sequence measurement, we changed the metric object of the above methods from a single image to an animation sequence. At the same time, we divided a weight based on the characteristics of the above indicators. We denote the measurement scores of the above three methods are s_p, s_s and s_f. The final animation score, which evaluate animation synthesis, is s and defined as

$$s = s_p + \alpha s_s + \beta \frac{1}{s_f}. \tag{10}$$

According to the importance of the animation sequence structure, we set the weight value α of s_s to 20. Meanwhile the FID score s_f is different from the other two method scores PSNR s_p and SSIM s_s. The closer a generative distribution is to the real distribution, the smaller the s_f value is. When the s_f value equals zero, it represents that the generative distribution is equal to the ground-truth distribution. Therefore, we set the weight value β for $1/s_f$ to 1 according to the quality score s_f of the currently generated distribution.

D. COMPARE WITH EXISTING WORK

We compare our method with the baseline MoCoGAN [11] and TGAN [24] using the metrics of evaluating animation which offered in the previous subsection. On the benchmark of Human Action [44], Our qualitative results show in Fig. 7 and measured value show in Tab. 4. The low value of FID means result better, and the minimum value of FID is 0. Conversely, the larger of the other three metrics, the better the results. The maximum value of PSNR is 255, and the maximum value of SSIM is 1. The minimum value of both are 0. Animation Score corrects the shortcomings of the three metrics through different weights (In this case $\alpha = 20$, $\beta = 1$).

To evaluate the metrics of PSNR and SSIM, we choose 10 consecutive frames generated from the two methods which comparing with related ground-truth frames. In addition, we compare 10 consecutive frames with 7,000 ground-truth frames to calculate the FID value. Finally, we combine the three metrics to calculate a relatively fair metric named animation score.

In baseline of MoCoGAN, the one type of recurrent neural networks, GRUs [19], have be used for sequence generation. We tested on the same device with 4 GPUs embedded, and the GPU model is Tesla V100 32GB SXM3. Compared to the baseline, which needs to input more animation sequences from dataset, our method only needs to input unordered images in the same dataset. From the evaluation results, our method is better than MoCoGAN when the time and computing resources are fixed.

We also compared our model with TGAN under the Moving-MNIST dataset. Some examples of results are shown in Fig. 8. The first two rows are the results of TGAN, and the last two rows are the corresponding results of our method. In TGAN, latent variables z, which is composed of z_1 and z_2, are slightly better than the latent variable only composing of z_1. Testing our model on the benchmark of Moving-MNIST, the numbers of dimensions are $n_d = 20$ and $n_c = 10$, are better than the numbers of dimensions $n_d = 20$ and $n_c = 20$. The overall effect of our method is better than TGAN. The specific evaluation can be seen in Tab. 5. We select 5 frames of each method result to measure PSNR and SSIM which compare with corresponding ground-truth frames. Meanwhile, we use these 5 frames to compare with 20,000 ground-truth frames to evaluate the FID score. The weights of the animation score are $\alpha = 20$ and $\beta = 1$.

This result also indirectly validates our empirical judgment on the effect of the pseudo-label dimensions, which shown in ablation study that the large-scale dataset needs more dimensions of pseudo-label to train a better result. At the same time, we can see in Fig. 8 that TGAN cannot generate sequences well in an unsupervised manner, and the problem can overcome by adding supervised labels. This is also different from our method by self-supervised learning.

V. DISCUSSION

A. COMPUTATIONAL ANALYSIS

By comparing different models, we found that adding effective calculations, such as adding the network layers and increasing the quality of training dataset, can improve the quality of animation synthesis. Upgrading the model structure

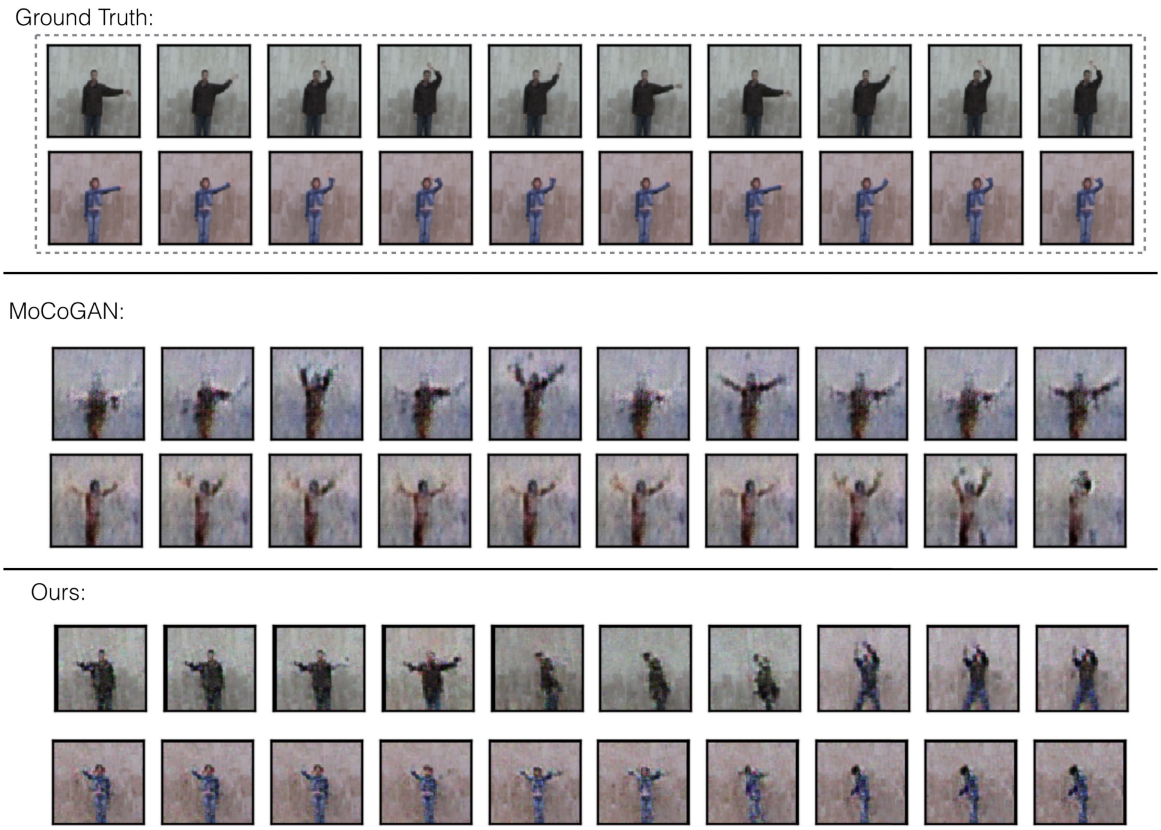

FIGURE 7. Qualitative results for image animation on the 2D Action dataset: MoCoGAN against our method. The first and second rows are the sequences from the ground-truth. The third and fourth rows are the sequences from the MoCoGAN output which similar with the ground-truth. The last two rows are the sequences from our method output which similar with the ground-truth. The input of MoCoGAN are motions and contents from the dataset. The input of our method does not need such motions. Our input are out-of-order motions extracted from the dataset [44].

TABLE 4. Animation comparisons with MoCoGAN on the benchmark of Human Action.

	MoCoGAN	Ours
PSNR (s_p)	17.36	**18.92**
SSIM (s_s)	0.54	**0.68**
FID (s_f)	423.87	**5.22**
Animation Score (s)	27.1	**33.29**

to improve computing efficiency is also a valuable task, especially when we only got limited computing resources.

As shown in the Tab. 6, we analyzed the computational consumption of models through its open code repository. Where TGAN [24] and MoCoGAN [11] require motions as supervised conditions, Monkey-GAN [10] and Pix2pix [1] require stronger supervision conditions, such as optical flow and semantic segmentation.

In the motions generation part, we observe the main computational networks in each model, the motions generator G_{motion}, and the animation discriminator $D_{animation}$ which contain mainly the computation of animation synthesis (motions and contents). For convenience, we abbreviate Conv2D and Deconv2D as C2D. Similarly, both Conv3D and Deconv3D are abbreviated as C3D.

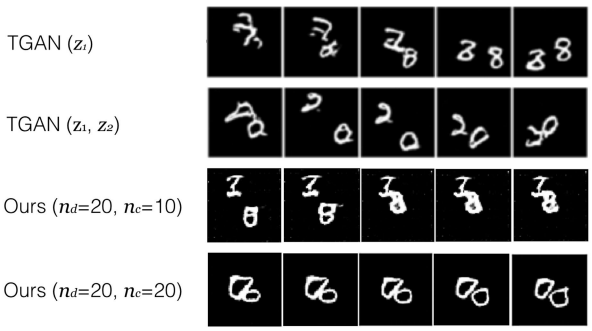

FIGURE 8. Qualitative results for animation synthesis on the Moving-MNIST: TGAN [24] against our method. The first and second rows come from TGAN. The third and fourth rows come from our method. The result shows that our method is much more consistent between continuous frames.

MoCoGAN needs to use GRU (one type of RNN) and C2D to complete the sequence generation training, and to use C3D to complete the discriminator training. Since each animation synthesized clips require more than 12 frames, it is empirically estimated that the training C3D requires 36-48 frames as input. So the calculation of C3D is 3-4 times higher than that of C2D which only calculates a single frame.

TABLE 5. Animation comparisons with TGAN on the benchmark of Moving-MNIST.

Method	PSNR / SSIM	FID	Animation Score
TGAN (z_1)	8.67 / 0.48	0.41	20.71
TGAN (z_1, z_2)	**9.32 / 0.56**	**0.33**	**23.52**
Ours ($n_c = 20, n_c = 10$)	9.39 / 0.54	0.77	21.49
Ours ($n_d = 20, n_c = 20$)	**10.17 / 0.64**	**0.12**	**31.30**

TABLE 6. Comparison of computational performance based on network architecture.

Model	Supervision	G_{motion}	$D_{animation}$
TGAN [24]	Motion	**C2D**	C3D
MoCoGAN [11]	Motion	RNN+C2D	C3D
Monkey-GAN [10]	Optical Flow	C3D	C3D
Pix2pix [1]	Segmentation	**C2D**	**C2D**
Ours	-	**C2D**	**C2D**

Normally, using C2D to replace the parts of C3D and RNN to train motions is the key ideal to reduce the amount of calculation in motions generation. In the case of insufficient computing resources and unsupervised conditions, our model is more effective than other high resolution animation synthesis models.

B. PSEUDO-LABELS WITH LATENT VARIABLES

In applied research that uses GAN as a generative model, high-dimensional latent variables as inputs, travel in a latent space, have always been an important research area. From simple one-hot labels to semantic segmentation labels, we can find the huge potential ability of latent variables to control the characteristics of GAN generation.

Under the pre-training model of a high resolution network [6], [7], we can add mapping models with embedding latent variables to control generative features. By changing the image features to generate videos, The ability of mapping features is related to the components of the networks and the dimensions of latent variables. To deal with this situation, StyleGAN added more fully connected layers to complete the features mapping. In this work we used a similar structure to enhance the capability of feature mapping. There are many possibilities for the mapping of latent variables to features. The latent variables with higher dimensions have more features that they can control. Our ablation study also prove this.

We attempt to discover how to use simple latent variables as much as possible to complete the mapping of animation features. In this paper, two types of pseudo-labels are learned to embedding in the latent variables, and then we map the latent variables into animated features. This is an effective way of animation synthesis by features mapping.

VI. CONCLUSION

In this paper we introduced a novel deep learning method for image synthesis. Motivated by InfoGAN and StyleGAN, we designed a new discriminator that can be trained to learn two types of pseudo-labels. In a self-supervised manner, our method allows a new generator to make pseudo-labels in the image dataset from latent variables. Our method can generate animations with the new generator by learning the pseudo-labels which can map the animation features by the discriminator.

In the experiment, we researched the relationship between the different numbers of pseudo-label dimensions and the performance of animation synthesis. Moreover, we studied controlling techniques for animations synthesis by the pseudo-labels of different dimensions and types. Furthermore, we compared our method with existing related methods through multiple metrics. Our experiments confirm the validity and feasibility of our proposed method.

Finally, we analyzed the computational efficiency from the network architecture and made a wider comparison. We discussed the relationship between pseudo-labels and latent variables, and the mapping relationship between latent variables and image features. In future work, we plan to extend our method on the more deep network architectures to generate higher quality animations and videos. Meanwhile, we will stand on the pre-training models of the current famous deep generative networks and explore the efficacy of pruning model for features mapping to generate higher quality animations and videos.

ACKNOWLEDGMENT

The authors would like to thank Yujun Shen (Ph.D. student at CUHK) and Timothy Jakobi (Ph.D. student at RMIT) for insightful discussions. They would also like to thank the help from the reviewers whose comments helped to shape this article.

REFERENCES

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976.

[2] C. Chan, S. Ginosar, T. Zhou, and A. Efros, "Everybody dance now," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5932–5941.

[3] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 105–114.

[4] T. R. Shaham, T. Dekel, and T. Michaeli, "SinGAN: Learning a generative model from a single natural image," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4569–4579.

[5] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–26.

[6] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.

[7] R. Abdal, Y. Qin, and P. Wonka, "Image2StyleGAN: How to embed images into the StyleGAN latent space?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4431–4440.

[8] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," 2018, *arXiv:1804.01523*. [Online]. Available: http://arxiv.org/abs/1804.01523

[9] X. Chen and W. Wang, "Uni-and-bi-directional video prediction via learning object-centric transformation," *IEEE Trans. Multimedia*, vol. 22, no. 6, pp. 1591–1604, Jun. 2020.

[10] A. Siarohin, S. Lathuiliere, S. Tulyakov, E. Ricci, and N. Sebe, "Animating arbitrary objects via deep motion transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2372–2381.
[11] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1526–1535.
[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[13] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[14] A. Van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, Jun. 2016, pp. 1747–1756.
[15] A. Van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with PixelCNN decoders," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA, 2016, pp. 4797–4805.
[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1×1 convolutions," in *Proc. NIPS*, 2018, pp. 10215–10224.
[17] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, *arXiv:1506.00019*. [Online]. Available: http://arxiv.org/abs/1506.00019
[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[19] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1724–1734.
[20] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, USA, vol. 28, no. 3, Jun. 2013, pp. 1310–1318.
[21] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.
[22] B. Chen, W. Wang, and J. Wang, "Video imagination from a single image with transformation generation," in *Proc. Thematic Workshops ACM Multimedia-Thematic Workshops*, New York, NY, USA, 2017, pp. 358–366.
[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.
[24] M. Saito, E. Matsumoto, and S. Saito, "Temporal generative adversarial nets with singular value clipping," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2849–2858.
[25] S. Wen, W. Liu, Y. Yang, T. Huang, and Z. Zeng, "Generating realistic videos from keyframes with concatenated GANs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2337–2348, Aug. 2019.
[26] S. Wen, W. Liu, Y. Yang, P. Zhou, Z. Guo, Z. Yan, Y. Chen, and T. Huang, "Multilabel image classification via Feature/Label co-projection," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Feb. 6, 2020, doi: 10.1109/TSMC.2020.2967071.
[27] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784
[28] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs created equal? A large-scale study," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 700–709.
[29] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA, 2017, pp. 5769–5779.
[30] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of GANs," 2017, *arXiv:1705.07215*. [Online]. Available: http://arxiv.org/abs/1705.07215
[31] J. Hyun Lim and J. Chul Ye, "Geometric GAN," 2017, *arXiv:1705.02894*. [Online]. Available: http://arxiv.org/abs/1705.02894
[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training GANs," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2234–2242.
[33] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, vol. 70, Aug. 2017, pp. 2642–2651.
[34] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA, 2016, pp. 2180–2188.
[35] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1302–1310.
[36] G. Balakrishnan, A. Zhao, A. V. Dalca, F. Durand, and J. Guttag, "Synthesizing images of humans in unseen poses," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8340–8348.
[37] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool, "Pose guided person image generation," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 406–416.
[38] A. Siarohin, E. Sangineto, S. Lathuiliere, and N. Sebe, "Deformable GANs for pose-based human image generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3408–3416.
[39] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 8026–8037.
[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.
[41] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3D face model for pose and illumination invariant face recognition," in *Proc. 6th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Sep. 2009, pp. 296–301.
[42] T. Xue, J. Wu, K. Bouman, and B. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 91–99.
[43] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Lille, France, 2015, pp. 843–852.
[44] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Oct. 2005, pp. 1395–1402.
[45] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 3730–3738.
[46] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6626–6637.
[47] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. 20th Int. Conf. Pattern Recognit.*, İstanbul, Turkey, Aug. 2010, pp. 2366–2369.
[48] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[49] S. Barratt and R. Sharma, "A note on the inception score," 2018, *arXiv:1801.01973*. [Online]. Available: http://arxiv.org/abs/1801.01973
[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

CHENG YU received the M.S. degree in technology of computer application from Northeastern University, Shenyang, China, in 2017. He is currently pursuing the Ph.D. degree in artificial intelligence with the Macau University of Science and Technology, Taipa, Macau. Since 2017, he has been a Lecturer with the Chongqing College of Electronic and Engineering, China. His research interests include computer vision, machine learning, and deep learning.

WENMIN WANG (Member, IEEE) received the Ph.D. degree in computer architecture from the Harbin Institute of Technology, China, in 1989. After then, he worked as an Assistant Professor and an Associate Professor with the Harbin University of Science and Technology and the Harbin Institute of Technology. Since 1992, he gained about 18 years of overseas industrial experiences in Japan and America, where he served as a Staff Engineer, the Chief Engineer, and the General Manager of software division. By the end of 2009, he worked with the School of Electronic and Computer Engineering, Peking University, China, as a Professor. Since 2019, he has been a Professor with the Macau University of Science and Technology. His current research interests include computer vision, multimedia retrieval, artificial intelligence, and machine learning.

JIANHAO YAN received the M.S. degree in statistical practice from Boston University, Boston, MA, USA, in 2019. He is currently pursuing the Ph.D. degree in artificial intelligence with the Macau University of Science and Technology, Taipa, Macau. His research interests include computer vision, multimedia retrieval, and deep learning.